AN ALGORITHM FOR OPTIMIZATION OF CERTAIN ALLOCATION MODELS

By

Paul Theodore Zmuida

United States
Naval Postgraduate School

THESIS

AN ALGORITHM FOR OPTIMIZATION
OF CERTAIN ALLOCATION MODELS

by

Paul Theodore Zmuida

Thesis Advisor: J. M. Danskin, Jr.

March 1971

Approved for public release; distribution unlimited.

T137757

An Algorithm for Optimization of Certain Allocation Models

by

Paul Theodore Zmuida
Major, United States Army
B.S. United States Military Academy, 1962

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
March 1971

Thesis
Z65
c 1

LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

ABSTRACT

Master's Thesis which discusses nature of allocation problems. Danskin Algorithm for solution of a convex function to be minimized over a closed convex set is developed. An example of application involving solution of a 3600 variable allocation problem using a computer is provided. Paper includes analysis of solution and discussion of problems encountered in computer application.

TABLE OF CONTENTS

I. INTRODUCTION

A large selection of problems faced within the military and commercial environments consist of finding the "best" way of using available resources. In most cases the "best" method of using these resources is that which provides the most desirous return to the using agency or organization. Typically, this return can be represented as a payoff associated with the use of resources, or in mathematical terms it is a function - the value of which is dependent on and determined by the amount or value of resources expended. Naturally, the availability and use of these resources are bounded and, therefore, act as a constraint on the using organization. This constraint could be that the organization is required to use some set minimum amount of a particular resource, or in the more usual sense only a given maximum amount of the resource is available for use. In the former case the resource is termed lower bounded, and in the latter case the resource is considered upper bounded. In many such problems the resource is both upper and lower bounded and usually some allocation of the resource between the bounds is acceptable.

If the situation exists where the bounds on a resource are known but where varying amounts of the resource can be expended in a number of different ways, then the manner in which the resource is allocated becomes of critical importance to the allocating organization in terms of the payoff it

receives from choosing a particular allocation scheme. By simple use of a crystal ball, random selection device, or some other means; and by exercising care not to violate the bounds, the organization can adequately allocate. However, it cannot be sure it has found the best allocation scheme.

If the payoff realized can be expressed as a mathematical function of the resources allocated, then the problem becomes susceptible to solution by mathematical programming methods. In many cases then the "best" or optimal allocation of resources can be found and proven to be optimal. These type problems can be described as "allocation" type problems. Typically, the value representation of a resource variable is a non-negative number less than one and indicating the percent of the total resource available which is to be expended in the particular manner or activity associated with the given variable.

The formulation of the payoff function is usually the deciding factor in determining what type of mathematical programming method could be used to find the optimal allocation or optimal solution to the problem. For instance, if the payoff function is linear in nature, then it may be possible to solve the problem by using existing linear programming methods. However, many payoff functions cannot be expressed as linear relationships even when attempts are made at reasonable approximations. In this case one must turn to other methods. In some cases it may be true that

no known mathematical method for optimizing exists. In this case it may be proper to reformulate the problem or possibly return to the crystal ball.

This paper will address a particular category of allocation problems, develop a method for solving problems which fit into this category, and provide a detailed example of a practical problem solved by the method developed. Credit for development of this method belongs to Dr. John M. Danskin, Professor - U. S. Naval Postgraduate School. This method, employing an algorithm termed the Direction Finding Algorithm, was presented by Dr. Danskin in various classroom lectures presented during his tenure at that school.

A. PRESENTATION OF THE PROBLEM

Consider an allocation type problem in which the payoff function is convex (or concave) and it is decided that the optimal solution is that solution which minimizes (or maximizes) the payoff function. Now, consider possible allocation of resources to be represented as $X = (x_1, x_2, \ldots x_n)$ where x_1 = percent of resources expended in activity #1, x_2 = percent of resources expended in activity #2, etc.; for $i = 1, \ldots, n$.

As indicated earlier, certain constraints must be heeded in finding the optimal solution. These constraints can be formulated as follows:

(1) It is necessary and desirable to use up all of the resources available,

(2) It may or may not be necessary to use some percent of the resource in any given activity,

(3) The percent of resource expended in activity i cannot exceed its upper or lower bounds, and

(4) Negative use of the resource is meaningless and not possible.

This problem as described can then be mathematically expressed as:

maximize $f(x)$ where $f(x)$ is concave

or

minimize $f(x)$ where $f(x)$ is convex

subject to constraints:

$$\sum_i x_i = 1$$

$$a_i \leq x_i \leq b_i$$

where a_i and b_i represent the upper and lower bounds respectively of the allocation to the i^{th} activity. It then follows that the following relationships are true:

$$0 \leq a_i < b_i$$

$$\sum_i a_i < 1 < \sum_i b_i \ .$$

Essential to the development and use of the Direction Finding Algorithm is the fact that the payoff or objective function must be differentiable. Therefore, this requirement that the gradient exists and can be calculated is considered as an added constraint.

This category of allocation problems just described is the category for which the about-to-be-developed Direction Finding Algorithm can be used to find the optimal solution.

II. DERIVATION AND DEVELOPMENT OF THE ALGORITHM

In this section the Direction Finding Algorithm will be developed and a step-by-step procedure described for its application.

Recall the original problem is to maximize (minimize) a concave (convex) differentiable function of the vector $X = (x_1, x_2, \ldots, x_n)$ with constraints as follows:

$$\sum_i x_i = 1 \qquad 0 \leq a_i < b_i$$

$$a_i \leq x_i \leq b_i \qquad \sum_i a_i < 1 < \sum_i b_i \; .$$

The basis of the algorithm is to find the direction of fastest increase. This means it is necessary to maximize the Directional Derivative which is defined in Ref. [1] as

$$D_\gamma f(x) = \sum_i \gamma_i f_{x_i}(x) = \sum_i \Omega_i \gamma_i$$

where Ω represents the gradient and γ represents the direction of the gradient.

To develop the algorithm, the following constraints are imposed on γ, the direction vector:

$$\sum_i \gamma_i = 0$$

$$\sum_i \gamma_i^2 = 1$$

$$\gamma_i \geq 0 \text{ if } x_i = a_i$$

$$\gamma_i \leq 0 \text{ if } x_i = b_i$$

The entire problem then becomes

$$\begin{aligned} \text{maximize} \quad & \Sigma \Omega \cdot \gamma \\ \text{subject to} \quad & \Sigma\, \gamma_i^2 = 1 \\ & \Sigma\, \gamma_i = 0 \\ & \gamma_i \geq 0 \text{ if } x_i = a_i \\ & \gamma_i \leq 0 \text{ if } x_i = b_i \end{aligned}$$

with the assumption that the maximum is non-negative.

The first step to solving this problem will be to show necessary conditions for solution by application of the well-known Kuhn-Tucker (K-T) Conditions which can be found in Ref. [2] and many other places in recent literature on non-linear programming. The constraints can be rewritten and LaGrange Multipliers assigned as follows:

CONSTRAINT	MULTIPLIER
$\Sigma\, \gamma_i^2 - 1 \leq 0$	λ_i'
$1 - \Sigma\, \gamma_i^2 \leq 0$	λ_i''
$\Sigma\, \gamma_i \leq 0$	μ_i'
$-\Sigma\, \gamma_i \leq 0$	μ_i''
$-\gamma_i \leq 0$ if $x_i = a_i$	ν_i
$\gamma_i \leq 0$ if $x_i = b_i$	π_i

Application of the Kuhn-Tucker Theorem then results in

$$\begin{aligned} \nabla (\Omega \cdot \gamma) = {} & \lambda' \nabla (\Sigma\, \gamma_i^2 - 1) + \lambda'' \nabla (1 - \Sigma \gamma_i^2) \\ & + \mu' \nabla (\Sigma\, \gamma_i) + \mu'' \nabla (-\Sigma\, \gamma_i) \\ & + \Sigma' (-\nu_i) \nabla (\gamma_i) + \Sigma'' (\pi_i) \nabla (\gamma_i) \end{aligned}$$

where

∇ as usual represents the gradient

Σ' implies summation over those i's for which $x_i = a_i$

Σ'' implies summation over those i's for which $x_i = b_i$.

If at this point only the i^{th} component of the gradient is considered, the above equation leads to

$$\Omega_i = \gamma_i^0(2\gamma_i' + 2\gamma_i'') + \mu_i' - \mu_i'' - \nu_i + \pi_i$$

where again ν_i enters consideration only if $x_i = a_i$ and π_i is considered only if $x_i = b_i$. The γ_i^0 represents the i^{th} component of the γ vector in the optimal solution as provided for in the Kuhn-Tucker Theorem. Letting $\lambda = 2\lambda' - 2\lambda''$ and $\mu = \mu' - \mu''$, the following results can readily be deduced from the previous equation and the original constraints:

$$\Omega_i = \begin{cases} \lambda\,\gamma_i^0 + \mu & \text{if } a_i < x_i < b_i \\ & \text{or if } x_i = a_i \text{ and } \gamma_i^0 > 0 \\ & \text{or if } x_i = b_i \text{ and } \gamma_i^0 < 0 \\ \lambda\,\gamma_i^0 + \mu - \nu_i & \text{if } x_i = a_i \text{ and } \gamma_i^0 = 0 \\ \lambda\,\gamma_i^0 + \mu + \pi_i & \text{if } x_i = b_i \text{ and } \gamma_i^0 = 0 \end{cases}$$

Since $\nu_i \geq 0$ and $\pi_i \geq 0$ from the Kuhn-Tucker Theorem, it is equivalently true that

$$\Omega_i \leq \lambda \gamma_i^0 + \mu \qquad \text{if } x_i = b_i \text{ and } \gamma_i^0 = 0 \qquad (1)$$

$$\Omega_i \geq \lambda \gamma_i^0 + \mu \qquad \text{if } x_i = a_i \text{ and } \gamma_i^0 = 0 \qquad (2)$$

$$\Omega_i = \lambda \gamma_i^0 + \mu \qquad \text{otherwise conditions satisfying the original constraints.} \qquad (3)$$

Therefore, if the optimal solution exists, then λ and μ exist and are non-negative and satisfy the above conditions.

The next step is to consider the sufficiency of the conditions. It is asserted that if λ and μ exist and λ is non-negative and if γ^0 satisfies results in (1), (2), and (3) above, then γ^0 maximizes $\Sigma\Omega\cdot\gamma$.

To prove this let γ be any arbitrary direction vector satisfying the conditions derived via the Kuhn-Tucker Theorem. Observe that $\Omega\cdot\gamma = \Sigma\Omega_i\gamma_i = (\Omega_i - \mu)\gamma_i$ since $\Sigma\gamma_i = 0$ from the original problem constraints. This is true for all i representing elements of the Ω and γ vectors. Therefore, the summation can be broken into three groups of terms as indicated in (1), (2), and (3) above; i.e.,

$$\Omega\cdot\gamma = \Sigma^{I}(\Omega_i-\mu)\gamma_i + \Sigma^{II}(\Omega_i-\mu)\gamma_i + \Sigma^{III}(\Omega_i-\mu)\gamma_i$$

where Σ^{I}, Σ^{II}, and Σ^{III} are summations over terms in (1), (2), and (3) respectively. $\Sigma^{I} \leq 0$ since $(\Omega_i-\mu) \leq 0$ from the K-T necessary condition (1) and $\gamma_i \geq 0$ from the original constraint. By similar reasoning it can be seen that $\Sigma^{II} \leq 0$. Therefore, it is true that $\Omega\cdot\gamma \leq \Sigma^{III}$.

However, $\Sigma^{III}(\Omega_i-\mu)\gamma_i = \lambda\Sigma^{III}\gamma_i^0\gamma_i$ from K-T condition (3) and $\lambda\Sigma^{III}\gamma_i^0\gamma_i = \lambda \sum_{\substack{\text{all} \\ i}} \gamma_i^0\gamma_i$ since $\gamma_i^0 = 0$ for all i not

contained in Σ^{III}, and $\lambda \sum_{\substack{\text{all} \\ i}} \gamma_i^0 \gamma_i \leq \lambda \left(\sum_{\substack{\text{all} \\ i}} \gamma_i^{0\,2} \right) \left(\sum_{\substack{\text{all} \\ i}} \gamma_i^2 \right) = \lambda$ from the Schwartz inequality and from the original constraints. Therefore, $\Omega \cdot \gamma \leq \lambda$, but $\Omega \cdot \gamma^0 = \Sigma(\Omega_i - \mu)\gamma_i^0 = \lambda \Sigma \gamma_i^{0\,2} = \lambda$. And since $\lambda \geq 0$, it is true that $\Omega \cdot \gamma \leq \Omega \cdot \gamma^0$ with the resultant fact that γ^0 maximizes $\Omega \cdot \gamma$ as was to be shown. Furthermore, the maximum value is λ or in terms of the sufficiency conditions; if λ can be found then the optimal solution is obtained.

At this point it is convenient to rewrite equation (3) and the conditions necessary for the equality:

$$\Omega_i = \lambda \gamma_i^0 + \mu \qquad \text{if } a_i < x_i < b_i$$

$$\text{or if } x_i = a_i \text{ and } \gamma_i^0 > 0$$

$$\text{or if } x_i = b_i \text{ and } \gamma_i^0 < 0.$$

Letting Σ' indicate a summation over the terms fulfilling these conditions and summing, results in

$$\Sigma' \Omega_i = \lambda \Sigma' \gamma_i^0 + \Sigma' \mu = \Sigma' \mu$$

or

$$\mu = \frac{1}{N'} \Sigma' \Omega_i \tag{4}$$

where N' is the total number of terms fulfilling the conditions. If the equation is squared first and then summed, the expression

$$\lambda = \Sigma' (\Omega_i - \mu)^2 \tag{5}$$

results. It also follows from substitution that

$$\gamma_i^0 = (\Omega_i - \mu)/\lambda. \tag{6}$$

With a method of calculating values for μ, λ, and γ_i^0 now in hand, development of the algorithm can proceed.

For notational ease, let DS = the "Distinguished Set" and be defined as that set of indices i such that $a_i < x_i < b_i$, or $x_i = a_i$ and $\gamma_i^0 > 0$, or $x_i = b_i$ and $\gamma_i^0 < 0$. In other words, the distinguished set represents those elements of the allocation vector, X, where the element does not lie on the boundary; or if the element is on the boundary, then the corresponding direction vector element implies a move away from the boundary. With this notation equation (4) can be rewritten as

$$\mu = \frac{1}{N_{DS}} \sum_{i \epsilon DS} \Omega_i \qquad (7)$$

where N_{DS} is the total number of elements in DS.

The elements contained in DS can be separated into three categories as follows:

Category #1. The X element is not on its boundary and there is no restriction on the γ element.

Category #2. The X element is on its lower boundary and the γ element is positive. From equation (6) then $\Omega_i > \mu$.

Category #3. The X element is on its upper boundary and the γ element is negative. From equation (6) then $\Omega_i < \mu$.

It is evident now that the algorithm for finding γ^0, or the direction of fastest increase, must search out and find each of the elements included in the three categories

above. Recall that μ is actually the average of all Ω_i for which $\gamma_i^0 \neq 0$. This means it is necessary to determine μ concurrently with determining the elements in DS. The following procedure provides the method for doing just this.

A. THE DIRECTION FINDING ALGORITHM

Step #1. Let $\mu = \frac{1}{N_{DS}} \sum_{i \in DS} \Omega_i$ and DS = $\{\phi\}$ to start.

Step #2. Place into DS each index i for which $a_i < x_i < b_i$. Note that this implies no restriction on γ_i. Note also that throughout the algorithm, once i has been entered into DS, it is no longer considered as a candidate for entry.

Step #3. If DS = $\{\phi\}$, then temporarily let $\mu = \min_i\{\Omega_i\}$. Do <u>not</u> place this i into DS (the minimum gradient element serves simply as a starting point). If DS $\neq \{\phi\}$, then calculate μ from equation (7).

Step #4. Search indices for cases where $x_i = b_i$. Find the smallest Ω_i associated with this search. If this $\Omega_i < \mu$, then enter this i into DS and recalculate μ. If this $\Omega \geq \mu$, then proceed to Step #6.

Step #5. Repeat Step #4 until no additional i's can be entered into DS.

Step #6. Search indices for cases where $x_i = a_i$. Find the largest Ω_i associated with this search. If this $\Omega_i > \mu$, then enter this i into DS and recalculate μ. If this $\Omega_i \leq \mu$, then proceed to Step #8.

Step #7. Repeat Step #6 until no additional i's can be entered into DS.

Step #8. Repeat Step #4 through Step #7 until no additional i's can be entered into DS.

Step #9. Calculate $\lambda = [\sum_{i \varepsilon DS} (\Omega_i - \mu)^2]^{\frac{1}{2}}$. If $\lambda = 0$ then STOP.

Step #10. Calculate γ^0 from equation (6).

This concludes the algorithm. At this point two special case considerations should be discussed:

CASE I. The possibility exists that only one element will be entered into the DS. In this event $\lambda = 0$ and the optimal solution lies on the boundary. γ^0 at this point indicates that any better solution would violate the original constraints.

CASE II. If more than one element has been entered into DS and $\lambda = 0$, then $\Omega_i = \mu$ for all i. In this event all Ω_i are obviously equal and the optimal solution has been reached. In fact, the value of λ is a measure of how close the procedure has come to the optimal solution. It may be true in a practical application of the algorithm that λ will diminish more slowly with each successive computation as the optimal solution in neared. It may be necessary in this case for the user to make a determination of the precision desired and to terminate calculations when a satisfactory degree of precision has been reached.

In the next section an example application of the algorithm in a practically oriented problem is provided.

III. EXAMPLE APPLICATION OF DIRECTION FINDING ALGORITHM

Consider a military oriented problem in which an ordnance delivering organization has responsibility for and the capability of firing its ordnance into a particular sector of territory. Assume that this organization has some means of assigning probabilities to the potential existence of a target at any of a number of specific locations within its sector of responsibility. Assume also that the organization knows what damage it can expect from a specific type round delivered at a specific point. Knowing this information, the organization would then like to know how to optimally distribute the inventory it intends to deliver so that the probability of target survival is minimized.[1]

As a first step to solution of this problem, the sector of responsibility is assumed to be square in geometric shape and an imaginary grid is imposed upon the sector. The individual squares of the grid are numbered sequentially beginning in the top left corner and working first across and then down. Location of a target (or the possible location of a target) can then be referenced by a specific numbered square of the grid. Expected damage in any grid square can be calculated from the number of rounds which impact in that

---

[1] This problem is formulated and presented for solution in a thesis paper prepared by Captain William A. Hesser, USMC, at the U. S. Naval Postgraduate School in February/March 1971.

square and/or in nearby squares. Allocation of ordnance is represented by the number of rounds delivered to a specific grid square.

The survival probability of a target, or the objective function which is to be minimized, can now be expressed as a mathematical equation

$$F(Y) = \sum_i p_i \exp\{-\sum_j \beta_{ij} y_j\}$$

where the variables have interpretations as follows:

p_i = probability that target is located in i^{th} square,

β_{ij} = probability of destruction of a target appearing in the i^{th} square by a round detonating in the j^{th} square (this is referred to as the damage function),

y_j = percent of total inventory to be expended allocated to the j^{th} square,

$F(Y)$ = survival probability of the target and is a function of the allocation of Y.

Note at this point that the survival function, $F(Y)$, is a convex function and the variable Y is actually a closed convex set (there are only a finite number of rounds which can be fired). Also note that in the trivial case when $Y = (0,0,\ldots,0)$, i.e., no firing takes place, the survival function reduces to $F(Y) = \sum_i p_i = 1$ which is appropriate.

The problem now is to determine what allocation of Y will minimize $F(Y)$.

It can be seen at this point that in order to obtain realistic and usable results, it is necessary to use a grid with a large number of squares. Correspondingly, the allocation vector is composed of a large number of elements.

The problem can now be represented as follows:

$$\text{minimize } F(Y) = \sum_i p_i \exp\{-\Sigma \beta_{ij} y_j\}$$

subject to constraints

$$\sum_i y_i = 1.0$$

and

$$y_i \geq 0.0.$$

From previous discussion and assuming the p_i and β_{ij} are known beforehand, this problem can be solved using the Direction Finding Algorithm. Observe that the first partial derivative required in the use of the algorithm is

$$F_{y_i}(Y) = \sum_i p_i \beta_{ij} \exp\{-\sum_j \beta_{ij} y_j\}.$$

Solution of this problem via the Direction Finding Algorithm was accomplished on the IBM 360 computer system at the U. S. Naval Postgraduate School. Programming was done in FORTRAN IV Language using the G-Level Compiler. A copy of the program listing is included after the appendices.

A. EXPLANATION OF EXAMPLE

Since the purpose of this example is to demonstrate application of the algorithm and not to solve a given existing problem, assignment of values to parameters of the problem was based on academic interest and convenience rather than practically oriented.

A grid size of 3600 squares, or equivalently a 60X60 matrix was used. It is felt this grid size selection is consistent with what might be used in a truly practical application and at the same time it enlarges the problem sufficiently so that information about machine time for calculation would be meaningful.

The distribution of probabilities (p_i's) used in this example was uniform in nature. A p = 1/3600 was assigned to each of the 3600 grid squares. In a normal application it is expected that most grid squares would have a p = 0.0 assigned to them and only a relatively small number would be assigned a positive probability. Those positive assignments would be most likely based on observer reports, known routes of movements, specific areas offering good cover and concealment, and other considerations such as these. However, in this example a uniform distribution was selected for two reasons. Firstly, making all p's positive is considered a worst case condition with respect to the number of individual calculations required and the resultant machine time necessary for execution. Since solution by the Direction Finding Algorithm is an iterative process, machine time is an important consideration and a known upper limit on this factor is extremely meaningful. Secondly, a uniform distribution was used so that effects on the boundaries of the grid could be observed. Since the damage function (β_{ij}'s) implies interaction between grid squares, it is not immediately obvious what the optimal allocation along the edges of the grid should be.

Selection of an appropriate damage function would be based on the type of weapon being employed and the result of statistical data obtained for that weapon. Inherently, the damage function selected directly affects total machine time necessary to solve the problem. However, this is a variable which is dependent on a specific situation and not of prime consideration in this paper. A simple exponential relationship was selected for the example because it was felt that an exponential function would be representative of most damage functions in terms of accuracy and machine time necessary for calculation.

Given the above explanation of problem parameters it now becomes necessary to develop a computer oriented logical approach to solution of the problem.

The logical sequence followed in finding the direction of fastest increase is explained earlier in this paper. At this point it should be noted that the program actually employs two Direction Finding Algorithms. In one case (SUBROUTINE DF$ALL in the program listing) the γ or direction element associated with each and every element of the allocation vector is calculated. This is necessary before a final optimal solution can be obtained. However, while working toward the optimal solution, a significant reduction in calculation time can be realized by finding the direction of fastest increase only on the variables which have a positive allocation currently associated with them or a positive γ in case the allocation is currently zero. For this reason

the program maintains a list (SUBROUTINE LIST) of those element indices which meet this criteria. With each iteration then, SUBROUTINE DF$LST is employed to find the direction of fastest increase only for those elements appearing on the "list." When the optimal solution is found for the current "list," execution of the program shifts to SUBROUTINE DF$ALL. After execution of DF$ALL, a new list is formed and the above outlined process is repeated. Only when optimization is indicated through the use of DF$ALL is execution terminated.

In order to employ the Direction Finding Algorithm as indicated, various other calculations obviously must be made. Other routines in the program are written to accomplish tasks such as evaluating the gradient, evaluating the objective function, and revising the current allocation to one which is closer to the optimal solution. These tasks must necessarily be carried out during each iteration of the solution process. A general flow chart outlining the logical procedure followed is depicted in Appendix A. A complete listing of subroutines employed along with a description of the purpose of each is provided in Appendix B.

Results obtained in the solution to this example problem are included. It is interesting to note in these results that the optimal solution has no allocation of resources in the rows or columns immediately adjacent to the edges of the grid. Obviously the damage resulting in these grid squares from rounds allocated to and impacting in nearby interior

grid squares is considered sufficient to compensate for loss of damage effect that would occur if rounds were allocated to the outermost rows and/or columns. The higher allocation in those nearby interior grid squares suggests this conclusion.

It is also interesting to note that the optimal solution is not completely symmetric as might be expected from using a uniform type distribution of the p_i's. Inspection of the final allocation shows the upper left corner of the array somewhat different from allocations assigned in the other three corners. This is the result of the starting allocation used in the solution process and the precision considered acceptable within the program. The initial starting feasible solution was for the entire inventory to be assigned to the extreme upper left grid square. This heavy allocation in just one location prevented a lesser positive allocation from appearing in nearby grid squares as the program stepped along towards optimality. The program was actually moving toward a totally symmetric final allocation, but settled for something less because of the precision cutoff programmed into the problem. If a higher degree of precision were demanded and if the program were rewritten to provide greater machine accuracy, the program would tend to a completely symmetric optimal solution.

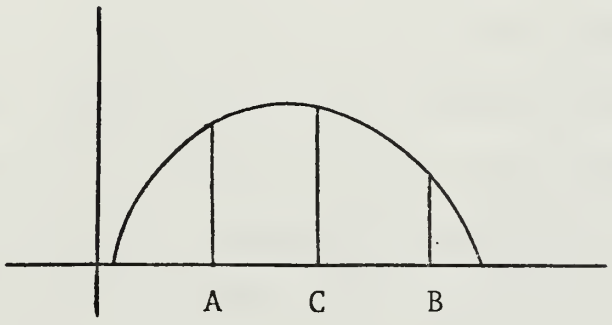
Certain difficulties were encountered in programming this problem for machine solution. A discussion of some of these difficulties and comments on observations is considered of academic interest to the reader at this point.

Since use of the Direction Finding Algorithm is an iterative process which moves from any feasible starting solution to a better one and then ultimately to an approximate optimal solution, selection of an initial starting point can be a critical matter. Initial attempts at solving this problem were done by assigning the entire inventory to the first grid square and proceeding from there. Analysis of results obtained from this "start from scratch" approach indicated that machine time in excess of twelve hours would be required to completely solve the problem. This required time was considered unacceptable and, therefore, an alternate approach was devised. The sector covered with a 3600 grid was initially considered to be covered with a 400 element grid. An optimal solution was found for the 400 grid case and this solution was then used as a starting solution for the 3600 grid case. Results in terms of machine time were remarkable. The optimal solution to the 400 case was obtained in 17.5 seconds of machine execution time. Overall machine time to the final optimal solution was six minutes nine seconds.

A point of interest in the program is the value of "D" or the distance which the allocation is moved in each iteration. Recall the lower and upper boundaries of the allocation elements are 0.0 and 1.0 respectively. The lower bound on D is 0.0 which is synonomous with no move at all. The upper bound on D is the maximum distance across the Y-space and is $\sqrt{2}$. In some calculations within the program D is

controlled by a variable which is near its boundary and trying to move in the direction of its boundary. The result of this situation in large problems such as this example is that D becomes very small and this leads to short moves on each iteration. When many variables are close to their boundaries and trying to move in that direction, the overall impact is a significant increase in machine execution time. However, there are various places in the solution process where D can be successfully reset to a predetermined value so that time is not wasted on relatively small moves when larger moves are permitted. There is also the problem of what to do with D when movement is made in a direction of increase but it has not resulted in an improved value of the objective function. To resolve these difficulties, an initial D value of 0.5 was selected. This had the effect of making the first step a large one. A reset value of 0.1 was decided upon and each time a new list was compiled, D was reset to this value. A modification procedure was used to adjust D when a move was made but an improvement in the objective function was not realized (rightly enough referred to as a "failure"). In this case D was simply cut in half and the move tried again. In the case of success in improving the value of the objective function, D was left alone and the same value used in the next iteration. The only exception to this procedure was in the case of a "success" which had been immediately preceded by a failure. In this circumstance D was halved

for the next iteration. To understand the rationale behind this consider the following simplified two-dimensional representation of a concave function which is to be maximized:



Let point A represent the current allocation and the direction of fastest increase is obviously to the right. Suppose the present D is equal in value to (B-A). When the move to B is attempted, a "failure" occurs since the value of the function is less than it was at point A. The procedure now calls for halving D and try to move again. This time the move is to point C and a "success" is realized. If D is not adjusted at this point, the next iteration would call for an attempted move back to point A which was the starting point for the previous iteration. To preclude this return to a previous allocation, D is cut in half after the occurrence of a success which had been immediately preceded by a failure. It should be noted that this procedure becomes particularly useful when the optimal solution is being neared and oscillations around the optimal solution start occuring.

Another consideration worthy of note is the influence of machine accuracy on the results of the problem. In a problem of this sort with potentially 3600 variables positive simultaneously, the effects of calculating, rounding, and cumulatively adding these variables must be taken into account. Early attempts at solution were made using single precision under FORTRAN IV. This approach was quickly changed to double precision for all real variables used in the program. Even so, results obtained were considered suspect in accuracy beyond 10^{-10}. To illustrate the problem, in this example the initial p_i value assignment to each grid square was made by assigning each grid square a value of 1/400. This value is supposed to be accurate in FORTRAN IV to 10^{-14}. As a check device the program then sums all p's assigned and prints the results. The resultant sum of 400 separate additions differs from 1.0 by 2.24×10^{-8}. Whether or not accuracy at this level is tolerable is up to the user, but most certainly is worthy of some consideration.

Machine storage space and execution time requirements in this problem were sufficiently significant to be worthy of comment. Because of the requirement to maintain a large number of large arrays of stored information (in the common storage region alone there are six arrays each of which contains 3600 double precision real numbers), this example program run on the IBM 360 System required 352,000 bytes of storage. Some space saving techniques were employed and,

undoubtedly, additional ones could be found. However, it should be recognized that the storage requirements are not a direct result of the methodology used in the Direction Finding Algorithm, but rather the result of the nature of the problem being solved. As mentioned previously, total execution time necessary was six minutes nine seconds. The approach of solving the problem on a small scale first and then using these results as a starting solution for the larger case was an instrumental factor in keeping execution time within reason. Other techniques involving the application of "clean coding" contributed toward reducing time requirements. One technique developed is considered to be of academic interest and involved the calculation of geometrical distance between squares. This calculation is necessary in evaluation of the gradient and in evaluating the function value. The distance calculation involves simply measuring the horizontal and vertical distance between the two grid squares in question and then evaluating the square root of the sum of squares. To illustrate the significance of this simple routine, it is done in excess of one million times in each iteration of the program. The execution time required to call the built-in machine square root function and evaluate the argument is known to be in excess of 200 microseconds. Since this obviously contributes to exorbitant machine execution time, it was decided to make the distance computation between any two grid squares once and for all at the beginning of the program and then store this information for later reference as required. This was

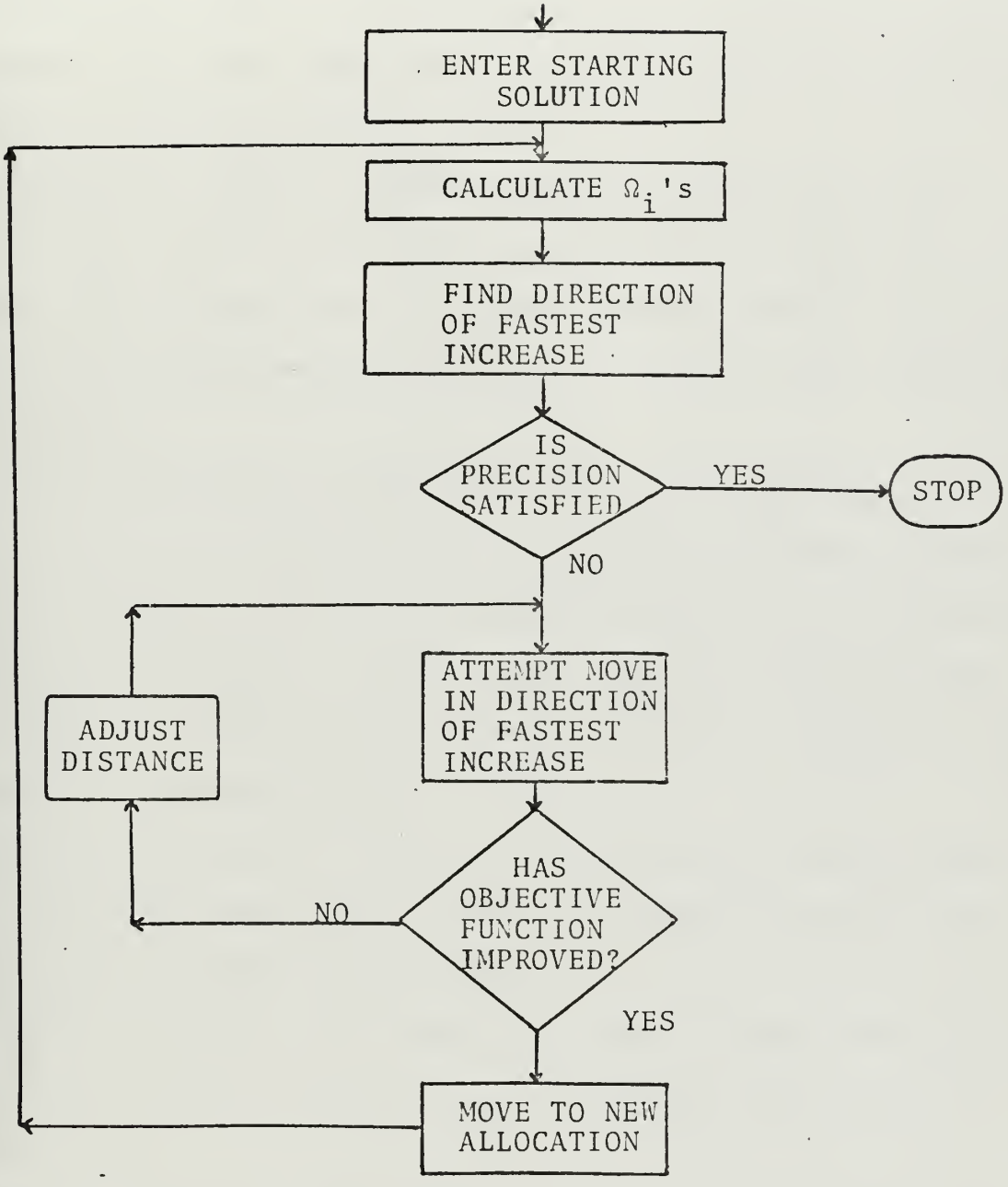
possible since the distance evaluations were somewhat repetitive in nature. The exact time saving incurred by application of this technique was not evaluated. However, a conservative estimate is that execution time was reduced by a factor of five and, therefore, makes the nominal additional storage requirement well worth while. The specific operation just described is contained in SUBROUTINE REFER.

IV. SUMMARY AND CONCLUSIONS

The Direction Finding Algorithm can be applied to a variety of concave or convex differentiable functions to find an optimal solution. However, the domain of these functions must be represented as a closed convex set. Its applicability in practical problems is particularly useful when the objective is to optimize allocation of resources.

In large scale problems it is easily adaptable to machine utilization. Time and storage requirements for a machine solution are primarily dependent on the nature of the specific problem being solved. The algorithm has relatively little effect on machine storage requirements. However, the algorithm's impact on machine time is the result of the boundary problem previously discussed. A method of overcoming the boundary problem as well as an extended application of this algorithm into an area of problems which require the allocated variable to be represented in typical matrix form rather than in vector form is currently being investigated by the aforementioned Dr. Danskin.

APPENDIX A: FLOWCHART-GENERAL LOGICAL PROCEDURE FOLLOWED



APPENDIX B: SUBROUTINE LISTING

SUBROUTINE AMMCAL: AMMCAL evaluates the objective function.

SUBROUTINE AREA: AREA determines which j's are associated with a given i in the β_{ij} term.

SUBROUTINE CHANGE: CHANGE converts the optimal solution obtained from the 400 grid case into the starting solution for the 3600 grid case. It also changes values of various parameters used within the program for use in the 3600 grid solution.

SUBROUTINE CNTROL: CNTROL controls the sequencing of events in solution of the problem. It tests each attempted move for success or failure and adjusts distance (D) accordingly.

SUBROUTINE DF$ALL: DF$ALL is the Direction Finding Algorithm which finds the direction of fastest increase for the entire allocation vector.

SUBROUTINE DF$LST: DF$LST is the Direction Finding Algorithm which finds the direction of fastest increase just for those variables on the list.

SUBROUTINE LIST: LIST compiles a list of those variables which are positive and/or those variables which have a positive γ.

SUBROUTINE MOVE: MOVE adjusts the allocation vector to reflect an attempted move in the direction of fastest increase. It moves to the boundary any variables which are within 10^{-14} of their boundary.

SUBROUTINE OMCALC: OMCALC calculates the vector of first partial derivatives of the allocation vector.

SUBROUTINE REFER: REFER fills a matrix with values representing geometrical distance between two grid squares.

SUBROUTINE SET$P: SET$P sets precision desired into the program and fills the P vector.

COMPUTER RESULTS

DISPLAY OF OPTIMAL ALLOCATION

NOTE 1: DISPLAY OF 20X20 (OR 400 GRID CASE) MATRIX IS SEPARATED INTO A LEFT HALF AND A RIGHT HALF DISPLAY.

NOTE 2: DISPLAY OF 60X60 (OR 3600 GRID CASE) MATRIX IS SEPARATED INTO TWELVE SECTORS AS FOLLOWS:

Y(1)	Y(11)	Y(21)	Y(31)	Y(41)	Y(51)
SECTOR I	SECTOR II	SECTOR III	SECTOR IV	SECTOR V	SECTOR VI
Y(1741)	Y(1751)	Y(1761)	Y(1771)	Y(1781)	Y(1791)
Y(1801)	Y(1811)	Y(1821)	Y(1831)	Y(1841)	Y(1851)
SECTOR VII	SECTOR VIII	SECTOR IX	SECTOR X	SECTOR XI	SECTOR XII
Y(3541)	Y(3551)	Y(3561)	Y(3571)	Y(3581)	Y(3591)

THE Y(***) INDICATES THE SUBSCRIPT NUMBER OF THE ALLOCATION VECTOR ELEMENT APPEARING IN THE INDICATED CORNER OF THAT SECTOR.

BELOW IS LEFT SIDE OF OPTIMAL SOLUTION OF 400 GRID CASE

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.00746	0.00654	0.00606	0.00592	0.00589
0.0	0.0	0.0	0.0	0.00462	0.00586	0.00485	0.00434	0.00419	0.00417
0.0	0.0	0.0	0.00462	0.00506	0.00456	0.00373	0.00333	0.00321	0.00319
0.0	0.0	0.00746	0.00586	0.00456	0.00370	0.00319	0.00296	0.00289	0.00287
0.0	0.0	0.00654	0.00485	0.00373	0.00319	0.00294	0.00285	0.00282	0.00281
0.0	0.0	0.00606	0.00434	0.00333	0.00296	0.00285	0.00282	0.00281	0.00281
0.0	0.0	0.00592	0.00419	0.00321	0.00289	0.00282	0.00281	0.00281	0.00281
0.0	0.0	0.00589	0.00417	0.00319	0.00287	0.00281	0.00281	0.00281	0.00281
0.0	0.0	0.00589	0.00417	0.00319	0.00287	0.00281	0.00281	0.00281	0.00281
0.0	0.0	0.00589	0.00417	0.00319	0.00287	0.00281	0.00281	0.00281	0.00281
0.0	0.0	0.00589	0.00417	0.00319	0.00287	0.00281	0.00281	0.00281	0.00281
0.0	0.0	0.00589	0.00417	0.00319	0.00287	0.00281	0.00281	0.00281	0.00281
0.0	0.0	0.00593	0.00422	0.00325	0.00294	0.00287	0.00287	0.00288	0.00288
0.0	0.0	0.00613	0.00448	0.00355	0.00325	0.00319	0.00319	0.00319	0.00319
0.0	0.0	0.00674	0.00529	0.00448	0.00422	0.00417	0.00417	0.00417	0.00417
0.0	0.0	0.00784	0.00674	0.00613	0.00593	0.00589	0.00589	0.00589	0.00589
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

BELOW IS RIGHT SIDE OF OPTIMAL SOLUTION OF 400 GRID CASE

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.00589	0.00589	0.00589	0.00589	0.00593	0.00613	0.00674	0.00784	0.0	0.0
0.00417	0.00417	0.00417	0.00417	0.00422	0.00448	0.00529	0.00674	0.0	0.0
0.00319	0.00319	0.00319	0.00319	0.00325	0.00355	0.00448	0.00613	0.0	0.0
0.00288	0.00288	0.00288	0.00287	0.00294	0.00325	0.00422	0.00593	0.0	0.0
0.00281	0.00281	0.00281	0.00281	0.00287	0.00319	0.00417	0.00589	0.0	0.0
0.00281	0.00281	0.00281	0.00281	0.00287	0.00319	0.00417	0.00589	0.0	0.0
0.00281	0.00281	0.00281	0.00281	0.00287	0.00319	0.00417	0.00589	0.0	0.0
0.00281	0.00281	0.00281	0.00281	0.00287	0.00319	0.00417	0.00589	0.0	0.0
0.00281	0.00281	0.00281	0.00281	0.00287	0.00319	0.00417	0.00589	0.0	0.0
0.00281	0.00281	0.00281	0.00281	0.00287	0.00319	0.00417	0.00589	0.0	0.0
0.00281	0.00281	0.00281	0.00281	0.00287	0.00319	0.00417	0.00589	0.0	0.0
0.00281	0.00281	0.00281	0.00281	0.00287	0.00319	0.00417	0.00589	0.0	0.0
0.00288	0.00288	0.00288	0.00287	0.00294	0.00325	0.00422	0.00593	0.0	0.0
0.00319	0.00319	0.00319	0.00319	0.00325	0.00355	0.00448	0.00613	0.0	0.0
0.00417	0.00417	0.00417	0.00417	0.00422	0.00448	0.00529	0.00674	0.0	0.0
0.00589	0.00589	0.00589	0.00589	0.00593	0.00613	0.00674	0.00784	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

NOW CONVERTING TO 3600 CASE

BELOW IS SECTOR I OF OPTIMAL SOLUTION

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00051
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00051
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00051
0.0	0.0	0.0	0.0	0.0	0.0	0.00083	0.00083	0.00083	0.00065
0.0	0.0	0.0	0.0	0.0	0.0	0.00083	0.00083	0.00083	0.00065
0.0	0.0	0.0	0.0	0.0	0.0	0.00083	0.00083	0.00083	0.00065
0.0	0.0	0.0	0.0	0.0	0.0	0.00073	0.00073	0.00073	0.00054
0.0	0.0	0.0	0.0	0.0	0.0	0.00073	0.00073	0.00073	0.00054
0.0	0.0	0.0	0.0	0.0	0.0	0.00073	0.00073	0.00073	0.00054
0.0	0.0	0.0	0.0	0.0	0.0	0.00067	0.00067	0.00067	0.00048
0.0	0.0	0.0	0.0	0.0	0.0	0.00067	0.00067	0.00067	0.00048
0.0	0.0	0.0	0.0	0.0	0.0	0.00067	0.00067	0.00067	0.00048
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00047
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00047
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00047
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046

BELOW IS SECTOR II OF OPTIMAL SOLUTION

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.00083	0.00083	0.00083	0.00073	0.00073
0.0	0.0	0.0	0.0	0.0	0.00083	0.00083	0.00083	0.00073	0.00073
0.0	0.0	0.0	0.0	0.0	0.00083	0.00083	0.00083	0.00073	0.00073
0.0	0.0	0.00051	0.00051	0.00051	0.00065	0.00065	0.00065	0.00054	0.00054
0.0	0.0	0.00051	0.00051	0.00051	0.00065	0.00065	0.00065	0.00054	0.00054
0.0	0.0	0.00051	0.00051	0.00051	0.00065	0.00065	0.00065	0.00054	0.00054
0.00051	0.00051	0.00056	0.00056	0.00056	0.00051	0.00051	0.00051	0.00041	0.00041
0.00051	0.00051	0.00056	0.00056	0.00056	0.00051	0.00051	0.00051	0.00041	0.00041
0.00051	0.00051	0.00056	0.00056	0.00056	0.00051	0.00051	0.00051	0.00041	0.00041
0.00065	0.00065	0.00051	0.00051	0.00051	0.00041	0.00041	0.00041	0.00036	0.00036
0.00065	0.00065	0.00051	0.00051	0.00051	0.00041	0.00041	0.00041	0.00036	0.00036
0.00065	0.00065	0.00051	0.00051	0.00051	0.00041	0.00041	0.00041	0.00036	0.00036
0.00054	0.00054	0.00042	0.00042	0.00042	0.00036	0.00036	0.00036	0.00033	0.00033
0.00054	0.00054	0.00042	0.00042	0.00042	0.00036	0.00036	0.00036	0.00033	0.00033
0.00054	0.00054	0.00042	0.00042	0.00042	0.00036	0.00036	0.00036	0.00033	0.00033
0.00048	0.00048	0.00037	0.00037	0.00037	0.00033	0.00033	0.00033	0.00032	0.00032
0.00048	0.00048	0.00037	0.00037	0.00037	0.00033	0.00033	0.00033	0.00032	0.00032
0.00048	0.00048	0.00037	0.00037	0.00037	0.00033	0.00033	0.00033	0.00032	0.00032
0.00047	0.00047	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00047	0.00047	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00047	0.00047	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031

BELOW IS SECTOR III OF OPTIMAL SOLUTION

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.00073	0.00067	0.00067	0.00067	0.00066	0.00066	0.00066	0.00066	0.00066	0.00065
0.00073	0.00067	0.00067	0.00067	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.00073	0.00067	0.00067	0.00067	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.00054	0.00048	0.00048	0.00048	0.00047	0.00047	0.00047	0.00046	0.00046	0.00046
0.00054	0.00048	0.00048	0.00048	0.00047	0.00047	0.00047	0.00046	0.00046	0.00046
0.00054	0.00048	0.00048	0.00048	0.00047	0.00047	0.00047	0.00046	0.00046	0.00046
0.00042	0.00037	0.00037	0.00037	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035
0.00042	0.00037	0.00037	0.00037	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035
0.00042	0.00037	0.00037	0.00037	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035
0.00036	0.00033	0.00033	0.00033	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00036	0.00033	0.00033	0.00033	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00036	0.00033	0.00033	0.00033	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00033	0.00032	0.00032	0.00032	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00033	0.00032	0.00032	0.00032	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00033	0.00032	0.00032	0.00032	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00032	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00032	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00032	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031

BELOW IS SECTOR IV OF OPTIMAL SOLUTION

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00036	0.00036	0.00036	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035	0.00035
0.00036	0.00036	0.00036	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035	0.00035
0.00036	0.00036	0.00036	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035	0.00035
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031

BELOW IS SECTOR V OF OPTIMAL SOLUTION

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.00066	0.00066	0.00066	0.00066	0.00066	0.00068	0.00068	0.00068	0.00075	0.00075
0.00066	0.00066	0.00066	0.00066	0.00066	0.00068	0.00068	0.00068	0.00075	0.00075
0.00066	0.00066	0.00066	0.00066	0.00066	0.00068	0.00068	0.00068	0.00075	0.00075
0.00046	0.00046	0.00047	0.00047	0.00047	0.00050	0.00050	0.00050	0.00059	0.00059
0.00046	0.00046	0.00047	0.00047	0.00047	0.00050	0.00050	0.00050	0.00059	0.00059
0.00046	0.00046	0.00047	0.00047	0.00047	0.00050	0.00050	0.00050	0.00059	0.00059
0.00035	0.00035	0.00036	0.00036	0.00036	0.00039	0.00039	0.00039	0.00050	0.00050
0.00035	0.00035	0.00036	0.00036	0.00036	0.00039	0.00039	0.00039	0.00050	0.00050
0.00035	0.00035	0.00036	0.00036	0.00036	0.00039	0.00039	0.00039	0.00050	0.00050
0.00032	0.00032	0.00033	0.00033	0.00033	0.00036	0.00036	0.00036	0.00047	0.00047
0.00032	0.00032	0.00033	0.00033	0.00033	0.00036	0.00036	0.00036	0.00047	0.00047
0.00032	0.00032	0.00033	0.00033	0.00033	0.00036	0.00036	0.00036	0.00047	0.00047
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046

BELOW IS SECTOR VI OF OPTIMAL SOLUTION

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.00075	0.00087	0.00087	0.00087	0.0	0.0	0.0	0.0	0.0	0.0
0.00075	0.00087	0.00087	0.00087	0.0	0.0	0.0	0.0	0.0	0.0
0.00075	0.00087	0.00087	0.00087	0.0	0.0	0.0	0.0	0.0	0.0
0.00059	0.00075	0.00075	0.00075	0.0	0.0	0.0	0.0	0.0	0.0
0.00059	0.00075	0.00075	0.00075	0.0	0.0	0.0	0.0	0.0	0.0
0.00059	0.00075	0.00075	0.00075	0.0	0.0	0.0	0.0	0.0	0.0
0.00050	0.00068	0.00068	0.00068	0.0	0.0	0.0	0.0	0.0	0.0
0.00050	0.00068	0.00068	0.00068	0.0	0.0	0.0	0.0	0.0	0.0
0.00050	0.00068	0.00068	0.00068	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0

BELOW IS SECTOR VII OF OPTIMAL SOLUTION

0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00046
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00047
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00047
0.0	0.0	0.0	0.0	0.0	0.0	0.00066	0.00066	0.00066	0.00047
0.0	0.0	0.0	0.0	0.0	0.0	0.00068	0.00068	0.00068	0.00050
0.0	0.0	0.0	0.0	0.0	0.0	0.00068	0.00068	0.00068	0.00050
0.0	0.0	0.0	0.0	0.0	0.0	0.00068	0.00068	0.00068	0.00050
0.0	0.0	0.0	0.0	0.0	0.0	0.00075	0.00075	0.00075	0.00059
0.0	0.0	0.0	0.0	0.0	0.0	0.00075	0.00075	0.00075	0.00059
0.0	0.0	0.0	0.0	0.0	0.0	0.00075	0.00075	0.00075	0.00059
0.0	0.0	0.0	0.0	0.0	0.0	0.00087	0.00087	0.00087	0.00075
0.0	0.0	0.0	0.0	0.0	0.0	0.00087	0.00087	0.00087	0.00075
0.0	0.0	0.0	0.0	0.0	0.0	0.00087	0.00087	0.00087	0.00075
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

BELOW IS SECTOR VIII OF OPTIMAL SOLUTION

0.00046	0.00046	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00036	0.00036	0.00036	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031
0.00046	0.00046	0.00035	0.00035	0.00035	0.00032	0.00032	0.00032	0.00031	0.00031
0.00047	0.00047	0.00036	0.00036	0.00036	0.00033	0.00033	0.00033	0.00032	0.00032
0.00047	0.00047	0.00036	0.00036	0.00036	0.00033	0.00033	0.00033	0.00032	0.00032
0.00047	0.00047	0.00036	0.00036	0.00036	0.00033	0.00033	0.00033	0.00032	0.00032
0.00050	0.00050	0.00039	0.00039	0.00039	0.00036	0.00036	0.00036	0.00035	0.00035
0.00050	0.00050	0.00039	0.00039	0.00039	0.00036	0.00036	0.00036	0.00035	0.00035
0.00050	0.00050	0.00039	0.00039	0.00039	0.00036	0.00036	0.00036	0.00035	0.00035
0.00059	0.00059	0.00050	0.00050	0.00050	0.00047	0.00047	0.00047	0.00046	0.00046
0.00059	0.00059	0.00050	0.00050	0.00050	0.00047	0.00047	0.00047	0.00046	0.00046
0.00059	0.00059	0.00050	0.00050	0.00050	0.00047	0.00047	0.00047	0.00046	0.00046
0.00075	0.00075	0.00068	0.00068	0.00068	0.00066	0.00066	0.00066	0.00066	0.00066
0.00075	0.00075	0.00068	0.00068	0.00068	0.00066	0.00066	0.00066	0.00066	0.00066
0.00075	0.00075	0.00068	0.00068	0.00068	0.00066	0.00066	0.00066	0.00066	0.00066
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

BELOW IS SECTOR IX OF OPTIMAL SOLUTION

0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00035	0.00035	0.00035	0.00035	0.00036	0.00036	0.00036	0.00036	0.00036	0.00035
0.00035	0.00035	0.00035	0.00035	0.00036	0.00036	0.00036	0.00036	0.00036	0.00036
0.00035	0.00035	0.00035	0.00035	0.00036	0.00036	0.00036	0.00036	0.00036	0.00036
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00065
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00065
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

BELOW IS SECTOR X OF OPTIMAL SOLUTION

0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031	0.00031
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032	0.00032
0.00036	0.00036	0.00036	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035	0.00035
0.00036	0.00036	0.00036	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035	0.00035
0.00036	0.00036	0.00036	0.00036	0.00036	0.00036	0.00035	0.00035	0.00035	0.00035
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046	0.00046
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066	0.00066
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

BELOW IS SECTOR XI OF OPTIMAL SOLUTION

0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00036	0.00036	0.00036	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00031	0.00031	0.00032	0.00032	0.00032	0.00035	0.00035	0.00035	0.00046	0.00046
0.00032	0.00032	0.00033	0.00033	0.00033	0.00036	0.00036	0.00036	0.00047	0.00047
0.00032	0.00032	0.00033	0.00033	0.00033	0.00036	0.00036	0.00036	0.00047	0.00047
0.00032	0.00032	0.00033	0.00033	0.00033	0.00036	0.00036	0.00036	0.00047	0.00047
0.00035	0.00035	0.00036	0.00036	0.00036	0.00039	0.00039	0.00039	0.00050	0.00050
0.00035	0.00035	0.00036	0.00036	0.00036	0.00039	0.00039	0.00039	0.00050	0.00050
0.00035	0.00035	0.00036	0.00036	0.00036	0.00039	0.00039	0.00039	0.00050	0.00050
0.00046	0.00046	0.00047	0.00047	0.00047	0.00050	0.00050	0.00050	0.00059	0.00059
0.00046	0.00046	0.00047	0.00047	0.00047	0.00050	0.00050	0.00050	0.00059	0.00059
0.00046	0.00046	0.00047	0.00047	0.00047	0.00050	0.00050	0.00050	0.00059	0.00059
0.00066	0.00066	0.00066	0.00066	0.00066	0.00068	0.00068	0.00068	0.00075	0.00075
0.00066	0.00066	0.00066	0.00066	0.00066	0.00068	0.00068	0.00068	0.00075	0.00075
0.00066	0.00066	0.00066	0.00066	0.00066	0.00068	0.00068	0.00068	0.00075	0.00075
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

BELOW IS SECTOR XII OF OPTIMAL SOLUTION

0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00046	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00047	0.00066	0.00066	0.00066	0.0	0.0	0.0	0.0	0.0	0.0
0.00050	0.00068	0.00068	0.00068	0.0	0.0	0.0	0.0	0.0	0.0
0.00050	0.00068	0.00068	0.00068	0.0	0.0	0.0	0.0	0.0	0.0
0.00050	0.00068	0.00068	0.00068	0.0	0.0	0.0	0.0	0.0	0.0
0.00059	0.00075	0.00075	0.00075	0.0	0.0	0.0	0.0	0.0	0.0
0.00059	0.00075	0.00075	0.00075	0.0	0.0	0.0	0.0	0.0	0.0
0.00059	0.00075	0.00075	0.00075	0.0	0.0	0.0	0.0	0.0	0.0
0.00075	0.00087	0.00087	0.00087	0.0	0.0	0.0	0.0	0.0	0.0
0.00075	0.00087	0.00087	0.00087	0.0	0.0	0.0	0.0	0.0	0.0
0.00075	0.00087	0.00087	0.00087	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
HERE IS THE MAIN PROGRAM
YY IS THE BASE POINT
J IS A POINTER TO ELEMENTS ON THE LIST
LONG IS THE TOTAL NUMBER OF ELEMENTS IN VECTORS
SIZE IS LENGTH OF ONE SIDE OF SQUARE GRID
FAIL1 IS TRUE IF SUCCESS FOLLOWS A FAILURE
SHORTD IS TRUE IF WORKING WITH A D CLOSE TO THE BOUNDARY
JJJ IS 1 IF ABBREVIATED OMCALC IS DESIRED
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
      COMMON Y(3600),YY(3600),J(3600),GA(3600),OM(3600),
     1IM(3600),P(3600),E(3600),DSTNCE(8,8),D,V2,AMM,AM,KO,
     1LONG,III,JJJ,JK,ISIZE,IRANGE,ILO,IHI,ITER,SHORTD,FAIL1
BLANK COMMON STATEMENTS IN SUBROUTINES HAVE BEEN OMITTED
FOR THIS PRINTING
      II=0
      III=0
      JJJ=0
      KKK=0
      KKKK=0
      NN=0
      ITER=0
      LONG=400
      ISIZE=20
      IRANGE=2
      ILO=3
      IHI=18
      AM=1.0
      D=0.1
      FAIL1=.FALSE.
      SHORTD=.FALSE.
      DO 1 I=1,3600
      Y(I)=0.0
      YY(I)=0.0
      OM(I)=0.0
      GA(I)=0.0
      IM(I)=0
      J(I)=0
 1    CONTINUE
FOLLOWING TWO CARDS INDICATE STARTING SOLUTION
      Y(1)=1.0
      YY(1)=1.0
      CALL SET$P
      CALL BEEP
      CALL CNTROL
      CALL CHANGE
      CALL CNTROL
AT THIS POINT PROBLEM IS SOLVED & NECESSARY WRITE
STATEMENTS NEED TO BE INSERTED
      STOP
      END


      SUBROUTINE CNTROL
THIS PORTION CONTROLS EXECUTION OF THE PROGRAM
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
 1000 CALL LIST
      CALL AMMCAL
FIRST TIME THROUGH AMMCAL IT IS CUT OFF SHORT
      CALL OMCALC
      CALL DESALL(Y,OM,GA,LONG,V2,&1090)
AT THIS POINT GET ON WITH NORMAL EXECUTION
 1010 CALL LIST
      D=0.1
      IF(III.EQ.0) D=0.5
      III=1
      FAIL1=.FALSE.
      SHORTD=.FALSE.
 1015 DO 1020 K=1,KO
      JK=J(K)
      YYY=YY(JK)
```

```
      GAA=GA(JK)
      IF(YYY+D*GAA.GE.0.0) GO TO 1020
      D=-YYY/GAA
      SHORTD=.TRUE.
 1020 CONTINUE
 1025 CALL MOVE
      CALL AMMCAL
      IF(AMM.LT.AM) GO TO 1040
BRANCH TO 1040 BELOW IF SUCCESS-OTHERWISE DO FOLLOWING:
      D=D/2
      FAIL1=.TRUE.
      GO TO 1025
IF SUCCESS,FIRST SET NEW AM & MOVE BASE POINT
 1040 AM=AMM
      DO 1045 K=1,KO
      JK=J(K)
 1045 YY(JK)=Y(JK)
      IF(FAIL1) GO TO 1050
      GO TO 1055
 1050 D=D/2
      FAIL1=.FALSE.
 1055 CALL OMCALC
      CALL DF$LST(Y,OM,GA,J,KO,LONG,V2,&1060)
      IF(.NOT.SHORTD) GO TO 1015
      D=0.1
      SHORTD=.FALSE.
      GO TO 1015
 1060 JJJ=1
HERE CALCULATE ONLY THOSE OMEGAS NOT ALREADY CALCULATED
      DO 1070 JJ=1,LONG
      IF(IM(JJ).EQ.1) GO TO 1070
      OM(JJ)=0.0
      JK=JJ
      CALL OMCALC
 1070 CONTINUE
      JJJ=0
      CALL DF$ALL(Y,OM,GA,LONG,V2,&1090)
      GO TO 1010
 1090 RETURN
      END


      SUBROUTINE SET$P
THIS ROUTINE ENTERS P VALUES AT START OF PROGRAM
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
      PREC=10.0**(-3)
      V=PREC/2**.5
      V2=V*V
 20   VAL=1.0/400.0
      DO 22 N=1,400
 22   P(N)=VAL
      SUMP=0.0
      DO 25 N=1,400
 25   SUMP=SUMP+P(N)
      DO 27 N=1,400
 27   P(N)=P(N)/SUMP
      RETURN
      END


      SUBROUTINE LIST
THIS IS LIST CONSTRUCTING SUBROUTINE
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
      K=0
      ITER=ITEP+1
      DO 60 JJ=1,LONG
      IF(Y(JJ).GT.0.0) GO TO 50
      IF(GA(JJ).GT.0.0) GO TO 50
      GO TO 60
 50   K=K+1
```

```
      J(K)=JJ
 60   CONTINUE
      KO=K
KO NOW CONTAINS LIST SIZE
      D=0.1
      RETURN
      END


      SUBROUTINE DF$LST($,OM$,GA$,J$,KO$,LONG$,V2$,*)
THIS IS DIRECTION FINDING ALGORITHM USING THE LIST
DF$LST ACCEPTS AS PARAMETERS THE FOLLOWING:
 $ IS VECTOR OF VARIABLE TO BE ALLOCATED
 OM$ IS VECTOR OF FIRST PARTIALS
 GA$ IS GAMMA VECTOR
 LONG$ IS CURRENT VECTOR LENGTH
 V2$ IS V*V WHERE V=PRECISION/2**.5
 J$ IS VECTOR OF POINTERS TO ELEMENTS ON THE LIST
 KO$ IS A NUMBER INDICATING THE LIST LENGTH
 * IS SPECIAL RETURN POINT (RETURN 1)
      IMPLICIT REAL*8(A-H,O-Z,$)
      REAL*8 LA,MU
      LOGICAL IND
      DIMENSION $(LONG$),OM$(LONG$),GA$(LONG$),J$(LONG$)
 THIS SUBROUTINE REQUIRES NO BLANK COMMON ACCESS
      COMMON/WORKA/IND(3600)
 100  N=0
      KK=0
      S=0.0
      DO 110 K=1,KO$
      JK=J$(K)
      GA$(JK)=0.0
      IND(JK)=.FALSE.
      IF($(JK).EQ.0.0) GO TO 110
      IF($(JK).EQ.1.0) GO TO 110
      IND(JK)=.TRUE.
      N=N+1
      S=S+OM$(JK)
 110  CONTINUE
      IF(N.EQ.0) GO TO 115
      MU=S/N
      AOM=MU
      GO TO 140
 115  AOM=OM$(J$(1))
      DO 120 K=1,KO$
      JK=J$(K)
      OMM=OM$(JK)
      IF(OMM.LT.AOM) AOM=OMM
 120  CONTINUE
      MU=AOM
 140  JJ=0
      DO 150 K=1,KO$
      JK=J$(K)
      IF(IND(JK)) GO TO 150
      IF($(JK).LT.1.0) GO TO 150
      IF(OM$(JK).GE.AOM) GO TO 150
      AOM=OM$(JK)
      JJ=JK
 150  CONTINUE
      IF(AOM.GE.MU) GO TO 170
      KK=0
      MU=(N*MU+AOM)/(N+1)
      AOM=MU
      N=N+1
      IND(JJ)=.TRUE.
      GO TO 140
 160  JJ=0
      DO 165 K=1,KO$
      JK=J$(K)
      IF(IND(JK)) GO TO 165
      IF($(JK).GT.0.0) GO TO 165
      IF(OM$(JK).LE.AOM) GO TO 165
```

```
      AOM=OM$(JK)
      JJ=JK
  165 CONTINUE
      IF(AOM.LE.MU) GO TO 180
      KK=0
      MU=(N*MU+AOM)/(N+1)
      AOM=MU
      N=N+1
      IND(JJ)=.TRUE.
      GO TO 160
  170 IF(KK.EQ.1) GO TO 190
      GO TO 160
  180 KK=1
      GO TO 140
  190 SS=0.0
      DO 195 K=1,KO$
      JK=J$(K)
      IF(.NOT.IND(JK)) GO TO 195
      SD=OM$(JK)-MU
      SS=SS+SD*SD
  195 CONTINUE
      IF(SS.LT.V2$) RETURN 1
      LA=DSQRT(SS)
      DO 200 K=1,KO$
      JK=J$(K)
      IF(.NOT.IND(JK)) GO TO 200
      SD=OM$(JK)-MU
      GA$(JK)=SD/LA
  200 CONTINUE
      RETURN
      END


      SUBROUTINE DF$ALL($,OM$,GA$,LONG$,V2$,*)
DF$ALL ACCEPTS AS PARAMETERS THE FOLLOWING:
 $ IS VECTOR OF THE VARIABLE TO BE ALLOCATED
 OM$ IS VECTOR OF FIRST PARTIALS
 GA$ IS GAMMA VECTOR
 LONG$ IS CURRENT VECTOR LENGTH
 V2$ IS V*V WHERE V=PRECISION/2**.5
 * IS SPECIAL RETURN POINT (RETURN 1)
      IMPLICIT REAL*8(A-H,O-Z,$)
      REAL*8 LA,MU
      LOGICAL IND
      DIMENSION $(LONG$),OM$(LONG$),GA$(LONG$)
THIS SUBROUTINE REQUIRES NO BLANK COMMON ACCESS
      COMMON/WORKA/IND(3600)
IND(I) IS SET TRUE WHEN $(I) ENTERS DS
  500 N=0
N IS COUNTER FOR NUMBER OF ELEMENTS IN DS
      KK=0
KK IS LOGIC SWITCH WHICH CONTROLS FLOW
      S=0.0
FIRST TO CHECK FOR ELEMENTS NOT ON THEIR BOUNDARIES
      DO 510 JJ=1,LONG$
      GA$(JJ)=0.0
      IND(JJ)=.FALSE.
      IF($(JJ).EQ.0.0) GO TO 510
      IF($(JJ).EQ.1.0) GO TO 510
      IND(JJ)=.TRUE.
      N=N+1
      S=S+OM$(JJ)
  510 CONTINUE
      IF(N.EQ.0) GO TO 515
      MU=S/N
AT THIS STAGE, MU IS AVERAGE OF INTERIOR ELEMENTS
      AOM=MU
      GO TO 540
IF DS IS EMPTY SET-FIND MINIMUM OMEGA
  515 AOM=OM$(1)
      DO 520 JJ=1,LONG$
      OMM=OM$(JJ)
```

```
      IF(OMM.LT.AOM) AOM=OMM
  520 CONTINUE
      MU=AOM
NOW TO CHECK FOR ELEMENTS ON UPPER BOUNDARY AND NOT IN DS
  540 JMARK=0
      DO 550 JJ=1,LONG$
      IF(IND(JJ)) GO TO 550
      IF($(JJ).LT.1.0) GO TO 550
      IF(OM$(JJ).GE.AOM) GO TO 550
      AOM=OM$(JJ)
      JMARK=JJ
  550 CONTINUE
      IF(AOM.GE.MU) GO TO 570
      KK=0
NOW TO RECALCULATE MU
      MU=(N*MU+AOM)/(N+1)
      AOM=MU
      N=N+1
      IND(JMARK)=.TRUE.
      GO TO 540
NOW TO CHECK FOR ELEMENTS ON LOWER BOUNDARY AND NOT IN DS
  560 JMARK=0
      DO 565 JJ=1,LONG$
      IF(IND(JJ)) GO TO 565
      IF($(JJ).GT.0.0) GO TO 565
      IF(OM$(JJ).LE.AOM) GO TO 565
      AOM=OM$(JJ)
      JMARK=JJ
  565 CONTINUE
      IF(AOM.LE.MU) GO TO 580
      KK=0
RECALCULATE MU AGAIN
      MU=(N*MU+AOM)/(N+1)
      AOM=MU
      N=N+1
      IND(JMARK)=.TRUE.
      GO TO 560
  570 IF(KK.EQ.1) GO TO 590
KK=1 IMPLIES NO MORE SEARCHING IS REQUIRED
      GO TO 560
  580 KK=1
      GO TO 540
NOW TO FIND SUM OF SQUARES OF ELEMENTS IN DS
  590 SS=0.0
      DO 595 JJ=1,LONG$
      IF(.NOT.IND(JJ)) GO TO 595
      SD=OM$(JJ)-MU
      SS=SS+SD*SD
  595 CONTINUE
      IF(SS.LE.V2$) RETURN 1
      LA=DSQRT(SS)
LAMBDA EQUALS SQUARE ROOT OF SUM OF SQUARES
      DO 600 JJ=1,LONG$
      IF(.NOT.IND(JJ)) GO TO 600
$(I) NOT IN DS IMPLIES GAMMA(I)=0
      SD=OM$(JJ)-MU
      GA$(JJ)=SD/LA
GAMMA VECTOR IS FILLED HERE
  600 CONTINUE
      RETURN
      END


      SUBROUTINE AMMCAL
THIS PART CALCULATES THE H'S AND AMM
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
      DIMENSION H(3600),Q(3600)
      COMMON/WORKB/H(3600),Q(3600)
      COMMON/XFER1/L,M,LL,LM,ML,MM
  300 DO 305 I=1,LONG
  305 H(I)=0.0
```

```
      DO 370 K=1,KO
      JK=J(K)
      IF(Y(JK).EQ.0.0) GO TO 370
      CALL AREA
      DO 360 LI=LL,LM
      DO 350 MI=ML,MM
      I=(LI-1)*ISIZE+MI
      IDL=IABS(LI-L)
      IDM=IABS(MI-M)
      IDL$=IDL+1
      IDM$=IDM+1
      S=DSTNCE(IDL$,IDM$)
      B=DEXP(-S)
      IF(LONG.EQ.3600) GO TO 341
      IF(S.GT.3.5) B=0.0
      GO TO 342
  341 IF(S.GT.10.5) B=0.0
  342 H(I)=H(I)+B*Y(JK)
  350 CONTINUE
  360 CONTINUE
  370 CONTINUE
      DO 375 I=1,LONG
      IF(P(I).EQ.0.0) GO TO 374
      Q(I)=DEXP(-H(I))
      E(I)=P(I)*Q(I)
      GO TO 375
  374 E(I)=0.0
  375 CONTINUE
      IF(III.EQ.0) RETURN
      S=0.0
      DO 380 I=1,LONG
  380 S=S+E(I)
      AMM=S
      RETURN
      END


      SUBROUTINE OMCALC
THIS ROUTINE CALCULATES THE OMEGAS
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
      COMMON/XFER1/L,M,LL,LM,ML,MM
  400 IF(JJJ.EQ.1) GO TO 420
      DO 410 I=1,LONG
      OM(I)=0.0
  410 IM(I)=0
      IF(III.NE.0) GO TO 415
      DO 490 JJ=1,LONG
      K=KO
      JK=JJ
      GO TO 420
  415 DO 490 K=1,KO
      JJ=LONG
      JK=J(K)
  420 CALL AREA
      IM(JK)=1
      DO 480 LI=LL,LM
      DO 470 MI=ML,MM
      I=(LI-1)*ISIZE+MI
      IF(P(I).EQ.0.0)GO TO 470
      IDL=IABS(LI-L)
      IDM=IABS(MI-M)
      IDL$=IDL+1
      IDM$=IDM+1
      S=DSTNCE(IDL$,IDM$)
      B=DEXP(-S)
      IF(LONG.EQ.3600) GO TO 441
      IF(S.GT.3.5) B=0.0
      GO TO 442
  441 IF(S.GT.10.5) B=0.0
  442 OM(JK)=OM(JK)+B*E(I)
  470 CONTINUE
```

```
480   CONTINUE
      IF(JJJ.EQ.1)RETURN
490   CONTINUE
      RETURN
      END

      SUBROUTINE AREA
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
      COMMON/XFER1/L,M,LL,LM,ML,MM
      L=(JK-1)/ISIZE+1
      M=JK-ISIZE*(L-1)
      IF(L.LT.ILO) GO TO 810
      LL=L-IRANGE
      GO TO 815
810   LL=1
815   IF(L.GT.IHI) GO TO 820
      LM=L+IRANGE
      GO TO 825
820   LM=ISIZE
825   IF(M.LT.ILO) GO TO 830
      ML=M-IRANGE
      GO TO 835
830   ML=1
835   IF(M.GT.IHI) GO TO 840
      MM=M+IRANGE
      GO TO 845
840   MM=ISIZE
845   RETURN
      END

      SUBROUTINE MOVE
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
30    DO 35 K=1,KO
      JK=J(K)
      YYY=YY(JK)
      GAA=GA(JK)
      Y(JK)=YYY+D*GAA
      IF(Y(JK).GT.1.0D-14) GO TO 34
      Y(JK)=0.0
      GO TO 35
34    IF(Y(JK).LT..99999999999999) GO TO 35
      Y(JK)=1.0
35    CONTINUE
      RETURN
      END

      SUBROUTINE REFER
THIS ROUTINE LOADS DSTNCE MATRIX
      IMPLICIT REAL*8(A-H,O-Z,$)
      LOGICAL SHORTD,FAIL1
70    DO 75 I=1,8
      DO 74 M=1,8
      I$=I-1
      M$=M-1
      SS=I$*I$+M$*M$
      S=DSORT(SS)
74    DSTNCE(I,M)=S
75    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CHANGE
THIS ROUTINE CONVERTS OPTIMAL SOLUTION FROM 400 CASE TO
STARTING SOLUTION FOR 3600 CASE
      IMPLICIT REAL*8(A-H,O-Z,$)
      INTEGER*4 BIGROW,BIGCOL
      LOGICAL SHORTD,FAIL1
      COMMON/WORKB/PP(3600),FILL(3600)
      D=0.1
      PREC=10.0**(-5)
      V=PREC/2**.5
      V2=V*V
      AM=1.0
      LONG=3600
      ISIZE=60
      IRANGE=7
      ILO=8
      IHI=53
      ITER=0
      III=0
      SHORTD=.FALSE.
      FAIL1=.FALSE.
      DO 13 I=1,3600
      P(I)=1.0/3600.0
      OM(I)=0.0
      GA(I)=0.0
      IM(I)=0
      J(I)=0
 13   CONTINUE
      DO 5010 I=1,400
FIRST DETERMINE ROW AND COLUMN OF INDEX NUMBER
      TEMPY=Y(I)/9.0
      NROW=(I-1)/20+1
      NCOL=I-20*(NROW-1)
NOW TO FIND BASE SQUARE WHICH IS TOP LEFT SQUARE
      BIGROW=NROW*3-2
      BIGCOL=NCOL*3-2
      IBASE=BIGCOL+60*(BIGROW-1)
      YY(IBASE)    =TEMPY
      YY(IBASE+1)  =TEMPY
      YY(IBASE+2)  =TEMPY
      YY(IBASE+60) =TEMPY
      YY(IBASE+61) =TEMPY
      YY(IBASE+62) =TEMPY
      YY(IBASE+120)=TEMPY
      YY(IBASE+121)=TEMPY
      YY(IBASE+122)=TEMPY
 5010 CONTINUE
NOW TO FILL Y & P MATRIX
      DO 5020 I=1,3600
 5020 Y(I)=YY(I)
      RETURN
      END
```

LIST OF REFERENCES

1. Danskin, J. M., Jr., The Theory of Max-Min, Springer-Verlag New York Inc., 1967.

2. Hillier, F. S. and Lieberman, G. J., Introduction to Operations Research, Holden-Day, Inc., 1967.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Professor J. M. Danskin, Jr. Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
4.	Department of Operations Analysis Naval Postgraduate School Monterey, California 93940	1
5.	MAJ Paul T. Zmuida, USA 26 Haven Street Schuylkill Haven, Pennsylvania 17972	1

DOCUMENT CONTROL DATA - R & D*

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
An Algorithm for Optimization of Certain Allocation Models

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Master's Thesis; March 1971

5. AUTHOR(S) (First name, middle initial, last name)
Paul T. Zmuida

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
March 1971	60	2

8a. CONTRACT OR GRANT NO	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT
Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

Master's Thesis which discusses nature of allocation problems. Danskin Algorithm for solution of a convex function to be minimized over a closed convex set is developed. An example of application involving solution of a 3600 variable allocation problem using a computer is provided. Paper includes analysis of solution and discussion of problems encountered in computer application.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Direction Finding Algorithm						

Thesis
Z65
c.1

126476

Zmuida
An algorithm for optimization of certain allocation models.

26 SEP 71
3 OCT 78
17 OCT 78

20678
24856
29384

Thesis
Z65
c.1

126476

Zmuida
An algorithm for optimization of certain allocation models.

thesZ65
An algorithm for optimization of certain
3 2768 000 98801 8
DUDLEY KNOX LIBRARY